Mem. S.A.It. Suppl. Vol. 14, 210

Memorie della
Supplementi

Supercomputing and stellar dynamics

R. Capuzzo-Dolcetta

Dipartimento di Fisica – Sapienza, Università di Roma, P.le A. Moro 2, I-00185 Roma, Italy
e-mail: roberto.capuzzodolcetta@uniroma1.it

Abstract. In this paper I outline some of aspects of modern celestial mechanics and stellar dynamics in the context of rapidly improving computing facilities. I highlight the great advantages in using, for astrophysical simulations, the modern, fast and inexpensive Graphic Processing Units (GPUs) acting as true supercomputers. Finally, I present and discuss some characteristics and performances of a new double-parallel code exploiting the joint power of multicore CPUs and GPUs.

Key words. Celestial mechanics– Stellar dynamics– Methods: N-body simulations– Supercomputing

1. Introduction

Gravity is a fundamental process in physics but its role is completely different in everyday terrestrial phenomena respect to that in astrophysics. On Earth, gravity is not too difficult to include because it simply acts as an external constant field that adds to other more complicated interactions among the constituents of the system under study. In other words, physical systems on Earth are not self-gravitating, and this provides an enormous simplification. In an astrophysical context, however, things are different: cosmic objects *are* self-gravitating, which largely determines their shapes, volumes, and dynamics, and often acts in conjunction with a gravitational attraction due to either external bodies or a general potential in which the object is embedded. External gravitational fields determine the orbit of an astronomical body, and influence its shape, at least at its periphery, by tidal interactions. A simple quantitative measure of the role of self-gravity to the whole energetics of a given system is the ratio, α, between the self-gravitational energy of the system and the energy given by the external gravitation field. For a typical terrestrial system, for instance the Garda lake, $\alpha \simeq 10^{-8}$, while for two, quite different, astronomical systems (a typical globular cluster in a galaxy and a typical galaxy in a galaxy cluster) $\alpha \simeq 10^{-2}$: a million times greater. Apart from the other, obvious, differences (a lake is composed of an incompressible liquid, where the collisional time scale is negligible respect to any other time scale in the system, while the globular cluster and the galaxy are composed by stars moving in volumes such that the collisional 2- body time scale is, in the case of globular cluster, comparable to, or, in the case of the galaxy, much longer than, the system orbital time and age), it is clear that while for the lake molecules mutual gravitational interactions are negligible respect to the external field, this is not so for the stars in globular clusters or galaxies.

Send offprint requests to: R. Capuzzo–Dolcetta

2. N-body systems in astrophysics

Since self-gravity cannot be neglected when studying the physics of astronomical objects, the dynamics of astrophysical systems is intriniecally difficult to study , even in the Newtonian approximation, because of the *double divergence* of the, simple, two-body interaction potential, $U_{ij} \propto 1/r_{ij}$, where r_{ij} is the euclidean distance between the i and j particle, $r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$.

The *ultraviolet divergence* arises from very close encounters, the *infrared divergence* arises because the gravitational interaction never vanishes. These divergences introduce a multiplicity of time scales (Aarseth 1985) and make it impossible to rely on statistical mechanics and/or to non-perturbative methods, as often done in other multibody physical problems. Actually, the Newtonian N-body dynamics is mathematically represented by the system of N second order differential equations

$$\begin{cases} \ddot{\mathbf{r}}_i = G \sum_{j=1, j\neq i}^{N} \frac{m_j}{r_{ij}^3} (\mathbf{r}_j - \mathbf{r}_i), \\ \dot{\mathbf{r}}_i(0) = \dot{\mathbf{r}}_{i0}, \\ \mathbf{r}_i(0) = \mathbf{r}_{i0}, \\ (i = 1, 2, \ldots, N). \end{cases} \tag{1}$$

This dynamical system is characterized by: (i) $O(N^2)$ complexity, (ii) strong nonlinearity, and, (iii), few constraints in the phase space. Sundman (1912) showed (without winning the King Oscar II Prize, already awarded to H. Poincarè...) that for the three-body problem there is a series solution for the coordinates in powers of $t^{1/3}$, convergent for all t except for initial data which correspond to zero angular momentum. This result was generalized to any N only in relatively recent times by Wang (1991). The power series solutions are so slow in convergence to be useless for practical use. Thus, the gravitational N-body problem must be approached numerically. The difficulties in doing this are both *theoretical* and practical. On the theoretical side, one has to face the chaotic behavior of the nonlinear system which is related to the extreme sensitivity of the systems differential equations to the initial conditions: a very small initial difference may result in an enormous change in the long-term behaviour of the system. Celestial dynamics presents one of the oldest examples of chaos in physics. This problem is almost unsolvable; it may be kept under some control by using sophisticated, high-order time integration algorithms. On the practical side, the greatest complication comes from the infrared divergence that requires computing all the N2 force interactions between the pairs in the systems. This results in an extremely demanding computational task, when N is large (see Table I). We will now discuss some of the problems arising when dealing with the numerical study of the evolution of self-gravitating systems over the astronomical range of N.

3. Small- and Large- N systems: from celestial mechanics to stellar dynamics

On the small- N side (N10, example: the solar system and extrasolar planetary systems) the problem is not that of enormous CPU time consumption, since the number of pairs is small, but the need for an the highest possible numerical precision to keep the roundoff error within acceptable bounds when integrating over many orbital times. In the few body problem, the point mass scheme (for instance, the Sun potential requires a multipole expansion) is not reliable enough and high precision codes are obligatory. Pair-wise force evaluation is computationally cheap due to the low number of pairs, but even very small roundoff errors increase secularly, time step by time step, making high-order symplectic integration algorithms unavoidable. The study requires a fast computer, capable of handling motion integration over a very extended time and able to evaluate forces with enormous precision.

Having discussed the few body regime, which is the realm of modern celestial mechanics and space dynamics, we now pass to the problem of intermediate and large-N-body systems, a task which is typical of the modern stellar dynamics. Force evaluation by pairs is computationally expensive, the mostly demanding part being the evaluation of the distance r_{ij} between the generic i and j particle. This requires the computation of a square root

Table 1. Some typical astronomical systems, with their star number (N), number of floating point operations needed for the force evaluations in a single system configuration (n_f) and CPU time required to the n_f operations by a single processor of 1 Gflops speed (t_{CPU}, in seconds). Note that 1.8×10^{14} sec $\simeq$ 5.7 Myr!

system	N	n_f	t_{CPU}
Open cluster	1000	1.5×10^7	0.02
Globular cluster	10^5	1.5×10^{11}	180
Galaxy	10^{11}	1.5×10^{23}	1.8×10^{14}

which, even with modern computers, is based on ancient methods e.g. Heros method, the Bombellis method, and the Newton-Raphson numerical solution of the quadratic equation $x^2 - r_{ij}^2 = 0$.

The single pair force evaluation requires about 30 floating point operations; so for an N-body system, $n_f = 30 \times N(N-1)/2$ floating point operations are required. A single processor (PE) with a speed of 1 Gflops can compute the single pair force in $\sim 3 \times 10^{-8}$ sec. Consequently, the whole N-star force computation requires the time indicated in Table 1 at every time step. Clearly, the task of numerically following the long term evolution of a large- N-body system by a code based on direct summation of pair forces is completely beyond the capability of even the highest performance computers. Currently, the profiling of any code for N-body integrations indicates that 70% of the CPU time is spent in the force evaluation.

What strategies must then be used?

The most natural approach is a combination of the following ingredients: (i) simplification of the interaction force calculation; (ii) reduction of the number of times that the forces have to be evaluated by a proper variation of the time step in both space and in time; and (iii) use of the most powerful (parallel) computers available. Points (i) and (ii) require deep numerical analysis, point (iii) requires the solution of the hard problem of parallelizing an N-body code. The simplification of a force calculation may be accomplished by introducing spatial grids for computing the large scale component of the gravitational force via the solution of the Poissons equation (with FastFourier codes, for example) and for the dynamic subdivision of the space domain with a recursive, octal tree to take computational advantage of a multipole expansion of the interaction potential (approach first used by Barnes & Hut (1986)). These are two means to reduce the particle-particle (PP) force evaluation to a particle-mesh (PM) or particle-particle particle- mesh (P3M), with obvious computational advantages (Hockney Eastwood (1988) for a general discussion). In addition to the complications introduced in the computer code, a clear limit of this procedure is the error introduced in the force evaluation, which can be reduced, over the small scale by keeping a direct PP force evaluation for close neighbours. Point (ii), time stepping variation, uses individual (per particle) time steps. Particles are advanced with a time step appropriate for the individual acceleration they experience, allowing a reduction in highly dynamical cases without stopping the overall calculation. Unfortunately, individual time stepping requires careful implementation to guarantee synchronous integration and often produces a reduction in the precision of the integration method. Finally, the parallelization of gravitational codes (point (iii)) is hard because the gravitational force felt by each particle depends on the position of all the others. This requires a difficult domain decomposition to realize a balanced computational weight for the various PEs of a parallel machine. In this context, we note that many groups choose to use 'dedicated' parallel architectures, which act as boosters of specific computations, such as the distances between particles. This is the route opened by the Japanese GRAPE group lead by Makino (Makino 1991). Another, intrigu-

ing, possibility is to use Graphic Processing Units (GPUs) as cheap alternatives to dedicated systems. GPUs are used to speed up force computations and give high computing performances at much lower cost, especially in cases where double precision is not required. This is the choice first explored in astrophysics by Portegies Zwart and his dutch group (Portegies Zwart, Belleman & Geldof 2007). Capuzzo–Dolcetta and collaborators in Italy (Capuzzo–Dolcetta, Maschietti & Mastrobuono–Battisti 2009) have constructed a direct N-body code implementing sophisticated 2-nd and 6-th order symplectic time-integration and using as force evaluation accelerator a pair of brand new NVIDIA TESLA C1060 Graphic processing Units (GPUs) programmed by means of the native NVIDIA Compute Unified Device Architecture (CUDA, see *www.nvidia.com/object/cuda_home.html*).

4. The NBSymple code

This code starts by generating the initial conditions for the N-body system whose individual masses may be selected from a given mass spectrum. The total mass of the system, M, gives the normalization factor. For the sake of simplicity, aiming first at checking the quality of integration and for testing the performance, particles were given an initial spatially uniform distribution within a sphere of given (unitary) radius, R, with velocities, also, uniformly distributed in direction and absolute values and rescaled to reproduce a given virial ratio, defined as $Q = 2K/|\Omega|$, where K and Ω are, respectively, the system kinetic and potential energies; for a stationary system, Q = 1. Note that setting $G = 1$ in the equations of motion means that the crossing time 'crossing' time $T = (GM)^{-1/2}R^{3/2}$ is the normalizing unit of time.

The code allows for the introduction of a softening parameter (ϵ) in the star-star interaction potential, usually taken as a fraction of the nearest neighbour average distance. The pairwise forces are summed with the force due to an external field, which is modeled by an analytical expression for the Galactic potential, e.g. Allen & Santillan (1991) (who decompose the Galactic potential into three components: a bulge, a disk and a halo. The bulge and the halo have a spherical symmetry, while the disk is axisymmetric.) Any generalization to diferent sets of initial conditions and external potentials is easily achieved using appropriate external subroutines provided by the user.

4.1. Time integration

It is well known that ordinary numerical methods for integrating the Newtonian equations of motions become dissipative and exhibit incorrect long term behaviour. This is a serious problem for N-body computations, particularly when studying their long term evolution. One possibility is to use symplectic integrators. Symplectic integrators are numerical integration schemes for Hamiltonian systems that conserve the symplectic two-form $d\mathbf{p} \wedge d\mathbf{q}$ exactly, so that $(\mathbf{q}(0),\ \mathbf{p}(0)) \rightarrow (\mathbf{q}(\tau),\ \mathbf{p}(\tau))$ is a canonical transformation. The transformation is characterized by time reversibility. If the integrator is not symplectic, the error of the total energy grows secularly, in general. Our code provides a choice of two different symplectic integrators. One is the simple, classic 'leapfrog' method, which is second-order accurate; the other is a more accurate, sixth-order, explicit scheme whose coefficients are taken from the first column of the Table 1 of Kinoshita, Yoshida & Nakai (1991), which leads to a time integration conserving energy much better than with the other two possible sets of coefficients in that Table. Of course, the 6-th order symplectic integrator is much slower than the leap frog, requiring 7 evaluations of force functions per time step, as e.g. in a 6-th order Runge Kutta method).

4.2. The computing platform

The workstation used to test and run our NBSymple code has a 2 Quad Core Intel Xeon processors, each running at 2.00GHz, 4GB DDR2 RAM at 667MHz and two NVIDIA TESLA C1060 GPUs, connected to the host via two slots PCI-E 16x.

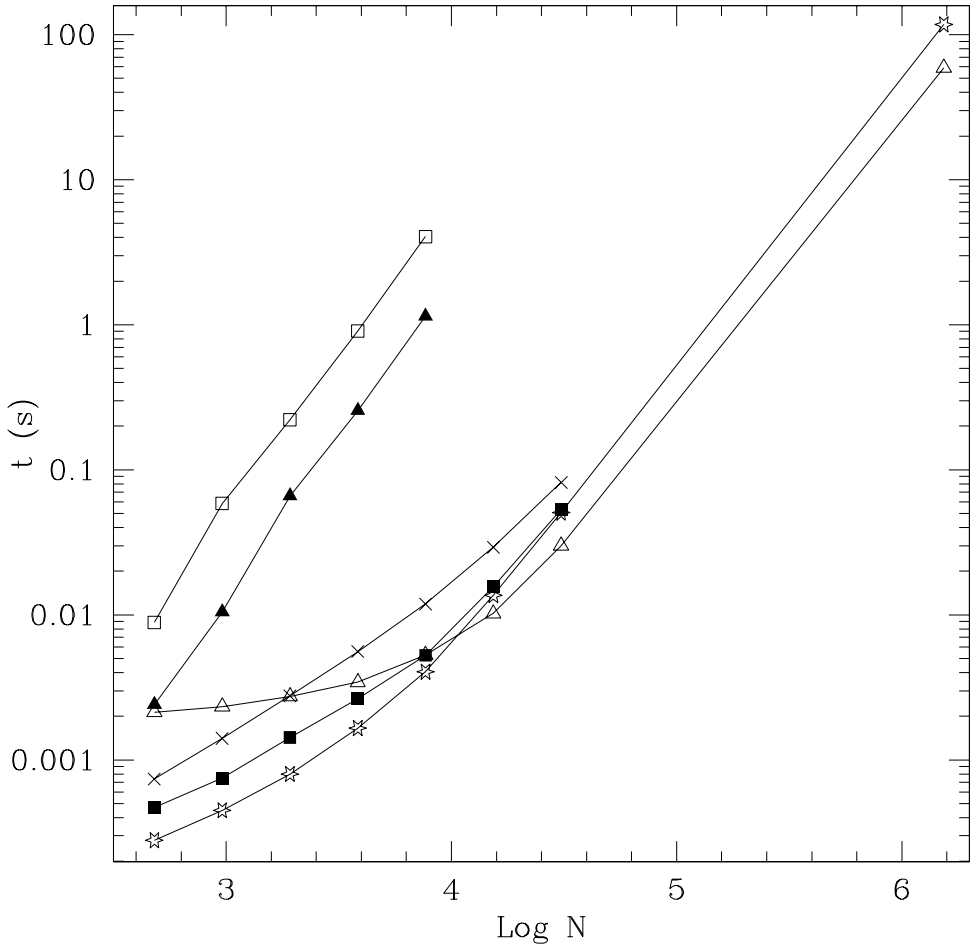

Fig. 1. The (averaged over 1000 cycles) solar time (in seconds) spent by various versions of NBSymple to perform a single integration step, as a function of N. Line with empty squares: NBSympleA code. Line with filled triangles: NBSympleB. Line with crosses: NBSympleC. Line with filled squares: NBSympleD. Line with empty triangles: NBSympleE with a single GPU. Line with stars: NBSympleE with two GPUs.

NVIDIA TESLA C1060 has 240 processors, each of them has a clock of 1.296GHz.

5. Results

Accurate testing of both the quality of the Nbody system integration and of the computational efficiency of NBSymple is given in the Capuzzo–Dolcetta, Maschietti & Mastrobuono–Battisti (2009) paper. In that paper, the various versions of the code are presented and discussed. Some versions work in single precision arithmetic (exploiting at best the GPU performances but not being fully satisfactory in terms of the precision) and in both harwdware (slower, more precise) and software (faster, less precise) double-precision arithmetic. The software double-precision is implemented following Goburov, Harfst & Portegies Zwart (2009).

The NBSymple code currently exists in 5 versions, each labeled with an alphabetic letter from A to E:

- NBSympleA: fully serial code running on a single Quad core processor;
- NBSympleB: single-parallel code which uses Open Multi-Processing (OpenMP) directives, for both the $O(N^2)$ pairwise interactions and the $O(N)$ calculations (i.e. the time integration and evaluation of the Galactic component of the force on the system stars) over the double Quad core host;
- NBSympleC: single-parallel code, where the $O(N^2)$ all-pairs interactions calculations are demanded to the NVIDIA TESLA C1060 GPU, using CUDA while all the remaining tasks are done by a single Quad core CPU;
- NBSympleD: double-parallel code, which again uses CUDA to evaluate the $O(N^2)$ portion of the code (as NBSympleC), while the $O(N)$ computations is parallelized sharing work between all the eight cores of the host, using OpenMP, as NBSympleB;
- NBSympleE: single-parallel code that uses CUDA on one or two GPUs to evaluate the total force over the system stars, i.e. both the all-pairs component and that due to the Galaxy.

We emphasize that the pairwise interactions evaluation are developed in the CUDA framework primarily following mIainly Nyland, Harris & Prins (2007). Here I just present a figure (Fig. 1) showing a comparison of the time (in seconds) spent by various versions of the NBSymple code for a single time step integration of an *N*body system as function of the number of bodies.

Acknowledgements. I am grateful to my collaborators D. Maschietti and A. Mastrobuono–Battisti, whose help was fundamental in developing the NBSymple code in the CUDA frame.

References

Aarseth, S., 1985, in Multiple time scales, edited by Brackbill, J.U. & Cohen B.J. (Academic Press, New York) pp.377 - 418

Allen, C., Santillan, A., 1991, RMexAA, 22, 255

Barnes, J. & Hut, P., 1986, Nature, 324, 446

Capuzzo–Dolcetta, R., Maschietti, D., Mastrobuono-Battisti, A., 2009, *in preparation*

Gaburov, E., Harfst, S., Portegies Zwart, S., 2009, NewA, 7, 630-637

Hockney, R.W., Eastwood J.W., 1988, Computer Simulation Using Particles (Hilger, Bristol)

Kinoshita, H., Yoshida, H., Nakai, H., 1991, CeMDA, 50, 59

Makino, J., 1991, PASJ, 43, 621

Nyland, L., Harris, M., Prins, J., 2007, Fast N-Body Simulation with CUDA, GPU Gems 3, H Nguyen, ed., (Prentice-Hall, New Jersey)

Portegies Zwart, S.F., Belleman, R. G., Geldof, P. M., 2007, NewA, 12, 641

Sundman, K.E., 1912, Acta Mathematica, 36, 105

Wang, Q., 1991, Cel. Mech. and Dyn. Astron., 50, 73.